We claim:

1. An improved manifold array (ManArray) processor architecture and instruction syntax comprising:

a regular instruction set;

means for executing the instructions of the instruction set; and

means for easily constructing a database for the instruction set.

2. The apparatus of claim 1 further comprising means for utilizing the database for the instruction set to readily create tools, such as assemblers, disassemblers, simulators or test case generators.

3. The apparatus of claim 1 further comprising:

means for parameterized test vectors; and

means for generating self-checking codes from the parameterized test vectors.

4. The apparatus of claim 1 further comprising means for parameterizing test vectors to create a parameterization; and

means for mapping the parameterization.

5. The apparatus of claim 1 wherein the regular instruction set is further defined in that each instruction has four parts delineated by periods with the four parts always in the same order to facilitate easy parsing by automated tools.

6. The apparatus of claim 1 wherein, every instruction has an instruction name; instructions that support conditional execution forms may have a leading (T. or F.) flag; arithmetic instructions may set a conditional execution state based on one of four flags (C=carry, N=sign, V=overflow, Z=zero); instructions that can be executed on both an SP and a PE or PEs specify the target processor via (.S or .P) designations, instructions

without an .S or .P designation are SP control instructions; arithmetic instructions always specify which unit or units that they execute on (A=ALU, M=MAU, D=DSU); load/store instructions do not specify which unit (all load instructions begin with the letter 'L' and all stores with the letter 'S'); arithmetic instructions (ALU, MAU, DSU) have data types to specify the number of parallel operations that the instruction performs (e.g., 1, 2, 4 or 8), the size of the data type (D=64 bit doubleword, W=32 bit word, H=16 bit halfword, B=8 bit byte, or FW=32 bit floating point) and optionally the sign of the operands (S=Signed, U=Unsigned); and load/store instructions have data types (D=doubleword, W=word, H1=high halfword, H0=low halfword, B0=byte0).

7. The apparatus of claim 1 wherein the database is organized as instructions with each instruction record in the database containing entries for conditional execution (CE), target processor (PROCS), unit (UNITS), datatype (DATATYPES) and operands for each datatype (FORMAT).

8. The apparatus of claim 7 wherein each instruction record further includes the number of cycles the instruction takes to execute (CYCLES), encoding tables for each field in the instruction (ENCODING) and configuration information (CONFIG) for subsetting the instruction set.

9. The apparatus of claim 1 further comprising an instruction-description data structure for an instruction.

10. The apparatus of claim 9 further comprising a second data structure defining input and output state for the instruction.

11. The apparatus of claim 1 further comprising means for generating multiple test vectors to set up and check state information for packed data type instructions.

12. An improved manifold array (ManArray) architecture and instruction syntax method comprising the steps of:

establishing a regular instruction set;

executing the instructions of the instruction set; and

constructing a database for the instruction set.

13. The method of claim 12 further comprising the step of utilizing the database for the instruction set to readily create tools, such as assemblers, disassemblers, simulators or test case generators.

14. The method of claim 12 further comprising the steps of:

parameterizing test vectors; and

generating self-checking codes from the parameterized test vectors.

15. The method of claim 12 further comprising the steps of:

parameterizing test vectors to create a parameterization; and

mapping the parameterization.

16. The apparatus of claim 12 further comprising the step of defining each instruction in the regular instruction set as having four parts delineated by periods with the four parts always in the same order to facilitate easy parsing by automated tools.

17. The apparatus of claim 12 further comprising the step of defining every instruction as having an instruction name; instructions that support conditional execution forms as having a leading (T. or F.) flag; utilizing arithmetic instructions to set a conditional execution state based on one of four flags (C=carry, N=sign, V=overflow, Z=zero); specifying for instructions that can be executed on both an SP and a PE or PEs the target processor via (.S or .P) designations, and defining instructions without an .S or .P designation as SP control instructions; specifying for arithmetic instructions which unit

or units that they execute on (A=ALU, M=MAU, D=DSU); not specifying for load/store instructions which unit (all load instructions begin with the letter 'L' and all stores with the letter 'S'); arithmetic instructions (ALU, MAU, DSU) having data types to specify the number of parallel operations that the instruction performs (e.g., 1, 2, 4 or 8), the size of the data type (D=64 bit doubleword, W=32 bit word, H=16 bit halfword, B=8 bit byte, or FW=32 bit floating point) and optionally the sign of the operands (S=Signed, U=Unsigned); and load/store instructions have data types (D=doubleword, W=word, H1=high halfword, H0=low halfword, B0=byte0).

18. The method of claim 12 further comprising the step of organizing the database as instructions with each instruction record in the database containing entries for conditional execution (CE), target processor (PROCS), unit (UNITS), datatype (DATATYPES) and operands for each datatype (FORMAT).

19. The method of claim 18 further comprising the step of establishing each instruction record as further including the number of cycles the instruction takes to execute (CYCLES), encoding tables for each field in the instruction (ENCODING) and configuration information (CONFIG) for subsetting the instruction set.

20. The method of claim 12 further comprising the step of establishing an instruction-description data structure for an instruction.

21. The method of claim 20 further comprising the step of establishing a second data structure defining input and output state for the instruction.

22. The apparatus of claim 12 further comprising the step of generating multiple test vectors to set up and check state information for packed data type instructions.